SEARCH METHODOLOGIES

SEARCH METHODOLOGIES
Introductory Tutorials in Optimization and Decision Support Techniques

Edited by

EDMUND K. BURKE
GRAHAM KENDALL

Edmund K. Burke
University of Nottingham
United Kingdom

Graham Kendall
University of Nottingham
United Kingdom

Library of Congress Control Number: 2005051623

ISBN-10: 0-387-23460-8
ISBN-10: 0-387-28356-0 (e-book)

ISBN-13: 978-0387-23460-1
ISBN-13: 978-0387-28356-2 (e-book)

Printed in the United States of America. Printed on acid-free paper.

9 8 7 6 5 4 3 2

springer.com

Contents

Foreword

This is not so much a foreword as a testimonial. As I embarked on the pleasant journey of reading through the chapters of this book, I became convinced that this is one of the best sources of introductory material on the search methodologies topic to be found. The book's subtitle, "Introductory Tutorials in Optimization and Decision Support Techniques", aptly describes its aim, and the editors and contributors to this volume have achieved this aim with remarkable success.

The chapters in this book are exemplary in giving useful guidelines for implementing the methods and frameworks described. They are tutorials in the way tutorials ought to be designed. I found no chapter that did not contain interesting and relevant information, and found several chapters that can only be qualified as delightful.

Those of us who have devoted a substantial portion of our energies to the study and elaboration of search methodologies often wish we had a simple formula for passing along the core notions to newcomers to the area. (I must confess, by the way, that I qualify as a relative newcomer to some of the areas in this volume.) While simplicity, like beauty, to some degree lies in the eyes of the beholder, and no universal or magical formula for achieving it exists, this book comes much closer to reaching such a goal than I would have previously considered possible. It will occupy a privileged position on my list of recommended reading for students and colleagues who want to get a taste for the basics of search methodologies, and who have an interest in equipping themselves to probe more deeply.

If good books may be likened to ladders that help us ascend to higher rungs of knowledge and understanding, we all know there are nevertheless many books written in technical areas that seem to be more like stumbling blocks, or at best broken stepping stools that deposit us in isolated nooks offering no clear access to continued means of ascent. Not so the present book. Its chapters lead to ideal sites for continued exploration, and offer compelling motivation for further pursuit of its ideas and frameworks. If my reckoning is not completely amiss, those who read this volume will find abundant reasons for shar-

ing my conviction that we owe its editors and authors a true debt of gratitude for putting this work together.

Fred Glover, Professor
Leeds School of Business, University of Colorado
Boulder, CO, USA

Preface

We first had the idea for this book over three years ago. It grew out of a one day workshop entitled, *Introductory Tutorials in Search, Optimization and Decision Support Methodologies (INTROS)*, which was held in Nottingham in August 2003. The aim of the workshop was to deliver basic introductions to a broad spectrum of search methodologies from across disciplinary boundaries. It was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) and the London Mathematical Society (LMS) and was attended by over one hundred delegates from all over the world. We were very fortunate to have eleven of the world's leading scientists in search methodologies presenting a range of stimulating and highly informative tutorials. All of the INTROS presenters have contributed to this volume and we have enhanced the content by inviting additional, specifically targeted, complementary chapters. We are pleased to be able to present such a comprehensive, multidisciplinary collection of tutorials in this crucially important research area.

We would like to take this opportunity to thank the many people who have contributed towards the preparation of this book. We owe a great debt of gratitude to the authors of the chapters. As one would expect from such a distinguished group of scientists, they have prepared their excellent contributions in a thoroughly reliable and professional manner. Without them, of course, the book would not have been possible. We are extremely grateful to our copy editor, Piers Maddox who excelled himself in bringing together, into one coherent structure, the various documents that we sent to him. We are also very grateful to Gary Folven, Carolyn Ford and their staff at Springer who have provided us with invaluable advice and support during every step of the way. We would like to offer our gratitude to Fred Glover for writing the foreword for the book. His warm praise is particularly pleasing. A special thank you should go to Emma-Jayne Dann and Alison Payne for all the administrative support they have given us, both in the preparation of this volume and in the organization of the INTROS workshop that underpinned it. We are also very thankful to EPSRC and LMS for the financial support they gave us to hold this workshop. Finally, we offer a special thank you to the INTROS delegates for their enthusiasm and their encouragement.

We hope you enjoy reading this volume as much as we have enjoyed putting it together. We are already planning a second edition and, if you have any comments which can help us improve the book, please do not hesitate to contact us. We would welcome your advice.

EDMUND K. BURKE AND GRAHAM KENDALL
ekb@cs.nott.ac.uk and gxk@cs.nott.ac.uk
June 2005

Chapter 1

INTRODUCTION

Edmund K. Burke, Graham Kendall
Automated Scheduling, Optimisation and Planning Research Group
The School of Computer Science and IT, University of Nottingham, UK

1.1 INTER-DISCIPLINARY DECISION SUPPORT: MOTIVATION

The investigation of search and optimization technologies underpins the development of decision support systems in a wide variety of applications across industry, commerce, science and government. There is a significant level of diversity among optimization and computational search applications. This can be evidenced by noting that a very small selection of such applications includes transport scheduling, bioinformatics optimization, personnel rostering, medical decision support and timetabling. More examples of relevant applications can be seen in Pardalos and Resende (2002), Leung (2004) and Dell'Amico et al. (1997). The exploration of decision support methodologies is a crucially important research area. The potential impact of more effective and more efficient decision support methodologies is enormous and can be illustrated by considering just a few of the potential benefits: more efficient production scheduling can lead to significant financial savings; higher quality personnel rosters lead to a more contented workforce; more efficient healthcare scheduling will lead to faster treatment (which could save lives); more effective cutting/packing systems can reduce waste; better delivery schedules can reduce fuel emissions.

This research area has received significant attention from the scientific community across many different academic disciplines. Indeed, a quick look at any selection of key papers which have impacted upon search, optimization and decision support will demonstrate that the authors have been based in a number of different *departments* including Computer Science, Mathematics, Engineering, Business, Management, and others. It is clearly the case that the investigation and development of decision support methodologies is inherently multi-disciplinary. It lies firmly at the interface of Operational Research and Artificial Intelligence (among other disciplines). However, not only is the underlying methodology inherently inter-disciplinary but the broad range of

application areas also cuts across many disciplines and industries. We firmly believe that scientific progress in this crucially important area will be made far more effectively and far more quickly by adopting a broad and inclusive multi-disciplinary approach to the international scientific agenda in this field. The way forward is inter-disciplinary.

This observation provides one of the key motivations for this book. The book is aimed primarily at first-year postgraduate students and final-year undergraduate students. However, we have also aimed it at practitioners and at the experienced researcher who wants a brief introduction to the broad range of decision support methodologies that is available in the scientific literature. In our experience, the key texts for these methodologies lie across a variety of volumes. This reflects the broad range of disciplines that are represented here. We wanted to bring together a series of entry-level tutorials, written by world-leading scientists from across the disciplinary range, in one single volume.

1.2 THE STRUCTURE OF THE BOOK

This book was originally motivated by the thought of being able to give first year Ph.D. students a single volume that would give them a basic introduction to the various search and optimization techniques that they might need to use during their research. In this respect the book can be read in a sequential manner. However, each chapter also stands alone and so the book can be dipped into when you come across a technique which you are not familiar with, or just need to find some general references on a particular topic.

If you want to read the book all the way through, we hope that the way we have ordered the chapters makes sense. We start by introducing (in Chapters 2 and 3) some classical search and optimization techniques which, although not always suitable (particularly when your problem has a very large search space), are still important to have in your "tool box" of methodologies. Many of the other chapters introduce various search and optimization techniques, some of which have been used for over 30 years (e.g. genetic algorithms, Chapter 4) and some which are relatively new (e.g artificial immune systems, Chapter 13). Some of the chapters consider some of the more theoretical aspects of search and optimization. The chapter by Darrell Whitley and John Paul Watson, for example, introduces *Complexity Theory and the No Free Lunch Theorem* (Chapter 11) whilst Colin Reeves considers *Fitness Landscapes* in Chapter 19.

One element of every chapter is a section called *Tricks of the Trade.* We recognize that it is sometimes difficult to know where to start when you first come across a new problem. Which technique or methodology is the most appropriate? This is a *very* difficult question to answer and forms the basis of much research in the area. It often requires experiments with a range of these

techniques. *Tricks of the Trade* is designed to give you some guidelines on how you should get started and what you should do, if you run into problems. Although tricks of the trade is towards the end of each chapter, we believe that it could be one of the first sections you read.

We have also included *Sources of Additional Information* in every chapter. These sections are designed as pointers to useful books, web pages, etc, which might be where you turn to next, once you have read the chapter in this book.

As this book is aimed primarily at the *beginner* (first-year Ph.D. student, final-year undergraduate, practitioner/researcher learning a new technique, etc) we thought it might be useful to explain some basic concepts which many books just assume that the reader already knows. Indeed, our own Ph.D. students and final-year undergraduates often make this complaint. We realize that the following list is not complete. Nor can it ever be, as we are not aiming to write a comprehensive encyclopedia. If you feel that any important terms are missing, please let the editors (authors of this introduction) know and we will consider including them in future editions. All of these concepts are, purposely, explained in an informal way so that we can get the basic ideas across to the reader. More formal definitions can be found elsewhere (see the *Sources of Additional Information* and *References*), including in the chapters of this book.

1.3 BASIC CONCEPTS AND UNDERLYING ISSUES

In this section we will go through a number of basic terms and issues and offer a simple description or explanation. In the spirit of attempting to explain these concepts to beginners, we will restrict the formal presentation of these concepts as much as possible. Instead, we will attempt to explain the basic ideas which underpin the terminology and the (often mathematical) formulations. Many of these terms are described and discussed throughout the book (see the index).

Artificial intelligence Artificial Intelligence is a broad term which can be thought of as covering the goal of developing computer systems which can solve problems which are usually associated with requiring human level intelligence. There are a number of different definitions of the term and there has been a significant amount of debate about it. However, the philosophical arguments about what is or is not Artificial Intelligence do not fall within the remit of this book. The interested reader is directed to the following (small) sample of general AI books: Negnevitsky (2005), Russell and Norvig (2003), Callan (2003), Luger (2002), MacCarthy (1996), Cawsey (1998), Rich and Knight (1991) and Nilsson (1998).

Operational research (Operations research) These two terms are completely interchangeable and are often abbreviated to OR. Different countries tend to use one or other of the terms but there is no significant difference. The field was established in the 1930s and early 1940s as scientists in Britain became involved in the *operational* activities of Britain's radar stations. After the war, the field expanded into applications within industry, commerce and government and spread throughout the world. Gass and Harris, in the preface to their excellent Encyclopedia of Operations Research and Management Science (Gass and Harris, 2001), present several definitions. However, as with Artificial Intelligence (above), we are not really concerned with the intricacies of different definitions in this book. The first definition they give says

> Operations Research is the application of the methods of science to complex problems arising in the direction and management of large systems of men, machines, materials and money in industry, business, government and defense.

This presents a reasonable summary of what the term means. For more discussion, and a range of definitions, on the topic, see Bronson and Naadimuthu (1997), Carter and Price (2001), Hillier and Lieberman (2005), Taha (2002), Urry (1991) and Winston (2004). For an excellent and fascinating early history of the field see Kirby (2003).

Management science This term is sometimes abbreviated to MS and it can, to all intents and purposes, be interchanged with OR. Definitions can be found in Gass and Harris (2001). However, they sum up the use of these terms nicely in their preface when they say

> Together, OR and MS may be thought of as the science of operational processes, decision making and management.

Feasible and infeasible solutions The idea of feasible and infeasible solutions is intuitive but let us consider the specific problem of cutting and packing, so that we have a concrete example which we can relate to. This problem arises in many industries: for example, in the textile industry where pieces for garments have to be cut from rolls of material, in the newspaper industry where the various text and pictures have to be laid out on the page and in the metal industry where metal shapes have to be cut from larger pieces of metal—see Dowsland and Dowsland (1992) for a more detailed review of this area. Of course, all these industries are different but let us consider a generic problem where we have to place a number of pieces onto a larger piece so that we can cut out the smaller pieces. Given this generic problem a feasible solution can be thought of as all the shapes being placed onto the larger sheet so that none of them overlap and all the pieces lie within the confines of the larger sheet. If some of the pieces overlap each other or do not fit onto the larger sheet, then the solution is infeasible. Of course, the problem definition is important when

considering whether or not a given solution is feasible. For example, we could relax the constraint that says that *all* of the shapes have to be placed on the larger sheet, as our problem might state that we are trying to cut out as many of the smaller shapes as possible, but it is not imperative that we include all the smaller pieces. A feasible solution is often defined as one that satisfies the *hard constraints* (see below).

Hard constraints For any given problem, there are usually constraints (conditions) that *have* to be satisfied. These are often called *hard constraints*. To continue with the cutting and packing example from above, the condition that no pieces can overlap is an example of a hard constraint. To take a new example, if we consider a nurse rostering problem, then an example of a hard constraint is the condition that no nurse can be allocated to two different shifts at the same time. If we violate a hard constraint, it leads to an *infeasible* solution. More information about research on nurse rostering problems can be seen in Burke et al. (2004).

Soft constraints and evaluation functions A *soft constraint* is a condition that we would like to satisfy but which is not absolutely essential. As an example, from nurse rostering again, we may have a soft constraint that says that we would like nurses to be able to express preferences about which shifts they would like to work. However, if this constraint is not fully met, a solution is still feasible. It just means that another solution which does meet the condition (i.e. more nurses have their personal working preferences met) would be of higher quality. Of course, there could be many competing soft constraints, which may provide a trade off in the *evaluation function* (measure of the quality of the solution which is also sometimes known as the *objective*, *fitness* or *penalty* function), as the improvement of one soft constraint may cause other soft constraint(s) to become worse. This is a situation where a multi-objective approach might be applicable (see Chapter 10).

Many problems have an evaluation function represented by a sum of each of the penalty values obtained for not satisfying each of the various constraints. Some problems simply ignore the hard constraints in the evaluation function and just disregard infeasible solutions. Another approach is to set a penalty value for the hard constraints but to set it very high so that any solution which violates the hard constraints is given a very high evaluation. A further possibility is to have dynamic penalties so that, at the start of the search, the hard constraints are given relatively low penalty values, so that the infeasible search space is explored. As the search progresses, the hard constraint penalty values are gradually raised so that the search eventually only searches the feasible regions of the search space.

Deterministic search This term refers to a search method or algorithm which always returns the same answer, given exactly the same input and starting conditions. Several of the methods presented in this book are not deterministic i.e. there is an element of randomness in the approach so that different runs on exactly the same starting conditions can produce different solutions. Note, however, that the term "*non-deterministic*" can mean something more than simply not being deterministic. See Chapter 11 for an explanation.

Optimization Within the context of this book, optimization can be thought of as the process of attempting to find the best possible solution amongst all those available. Therefore, the task of optimization is to model your problem in terms of some evaluation function (which represents the quality of a given solution) and then employ a search algorithm to minimize (or maximize, depending on the problem) that objective function. Most of the chapters in this book are describing methodologies which are aiming to optimize some function. However, most of the problems are so large that it is impossible to guarantee that the solution obtained is the optimal one. The term optimization can lead to confusion because it is sometimes also used to describe a process which returns the guaranteed optimal solution (which is, of course, subtly different from the process which just aims to find the best solution possible).

Local and global optimum Figure 1.1 illustrates the difference between a local and global optimum. A local optimum is a point in the search space where all neighboring solutions are worse than the current solution. In Figure 1.1, there are four local optima. A global optimum is a point in the search space where *all* other points in the search space are worse than (or equal to) the current one.

Exhaustive search By carrying out an exhaustive search, you search every possible solution and return the optimal (best) one. For small problems, this is an acceptable strategy, but as problems become larger it becomes impossible to carry out an exhaustive search. The types of problem that often occur in real world search and optimization problems tend to grow very large very quickly. We will illustrate this by considering a very well known problem: the traveling salesman problem (often referred to as TSP). This can be thought of as the problem of attempting to minimize the distance taken by a traveling salesman who has to visit a certain number of cities exactly once and return home. See Johnson and McGeoch (1997) or Lawler et al. (1990) for more details about the TSP. With a very small number of cities, the number of possible solutions is relatively small and a computer method can easily exhaustively check all possibilities (the search space) and return the best one. For example, the problem with five cities has a search space of size 12. So all 12 possibilities can

be very easily checked. However, for a 50-city problem (10 times the number of cities), the number of solutions rises to about 10^{60}. Michalewicz and Fogel, in their excellent book on modern heuristics, consider exactly this 50 city problem. They say,

> There are only 1,000,000,000,000,000,000,000 [10^{20}] liters of water on the planet so a 50-city TSP has an unimaginably large search space. Literally, it's so large that as human, we simply can't conceive of sets with this many elements.
>
> (Michalewicz and Fogel, 2004)

Therefore, for large problems (and large does not have to be that large), an exhaustive search is simply not an option. However, even if it is a possibility (i.e. the search space is small enough to allow us to carry out an exhaustive search) we must know how to systematically navigate the search space. This is not always possible.

Complexity This term refers to the study of how difficult search and optimization problems are to solve. It is covered in Chapter 11.

Order (Big O notation) This term and an associated notation is used at various places in this book and so we define it here. Suppose we have two functions $f(x)$ and $g(x)$ where x is, of course, a variable. We say that $g(x)$ is of the order of $f(x)$ written $g(x) = \mathrm{O}(f(x))$ if, for some constant value K, $g(x) \leq Kf(x)$ for all values of x which are greater than K. This notation is often used when discussing the time complexity of search algorithms. In a certain sense, $f(x)$ *bounds* $g(x)$ once the values of x get beyond the value of K.

Heuristics When faced with the kind of problem discussed in the exhaustive search section above, we have to accept that we need to develop an approach to obtain high-quality solutions—but optimality cannot be guaranteed (without *checking out* all the possibilities). Such an approach is called a heuristic. The following two definitions provide good descriptions.

> A heuristic technique (or simply heuristic) is a method which seeks good (i.e. near-optimal) solutions at a reasonable computation cost without being able to guarantee optimality, and possibly not feasibility. Unfortunately, it may not even be possible to state how close to optimality a particular heuristic solution is.
>
> (Reeves, 1996)

> A "rule of thumb" based on domain knowledge from a particular application, that gives guidance in the solution of a problem.... Heuristics may thus be very valuable most of the time but their results or performance cannot be guaranteed.
>
> (Oxford Dictionary of Computing, 1996)

There are many heuristic methods available to us. Some examples are simulated annealing (Chapter 7), genetic algorithms (Chapter 4), genetic programming (Chapter 5) and tabu search (Chapter 6). The term "approximate" is

sometimes used in connection with heuristic methods but it is important not to confuse with approximation methods (see Chapter 18).

Constructive heuristics Constructive heuristics refer to the process of building an initial solution from scratch. Take university examination timetabling as an example (Burke and Petrovic, 2002; Petrovic and Burke, 2004; Schaerf, 1999). One way to generate a solution is to start with an empty timetable and gradually schedule examinations until they are all timetabled. The order in which the examinations are placed onto the timetable is often important. Examinations which are more difficult to schedule (as determined by a heuristic measure of difficulty) are scheduled first in the hope that the *easier* examinations can *fit around* the difficult ones.

Constructive heuristics are usually thought of as being fast as they are often a single-pass approach.

Local search heuristics Local search can be thought of as a heuristic mechanism where we consider *neighbors* of the current solution as potential replacements. If we accept a new solution from this neighborhood, then we *move* to that solution and then consider its neighbors (see hill climbing (below) for some initial discussion of this point). What we mean by *neighbor* is dependent upon the problem solving situation that we are confronted with. Some of the techniques presented in this book can be described as local search methods. For example, see simulated annealing (Chapter 7) and tabu search (Chapter 6). Hill climbing is also a local search method (see below). For more information about local search see Aarts and Lenstra (1997). Note the difference between a constructive heuristic which builds a solution from scratch and a local search heuristic which moves from one solution to another. It is often the case that a constructive heuristic is used to generate a solution which is employed as the starting point for local search.

Hill climbing Hill climbing is probably the most basic local search algorithm. It is easy to understand and implement but suffers from getting stuck at a local optimum (see below). In the following discussion, we will assume we are trying to maximize a certain value. Of course, minimizing a certain value is just an analogous problem, but then we would be *descending* rather than *climbing*.

The idea behind hill climbing is to take the current solution and generate a neighbor solution (see local search) and move to that solution only if it has a higher value of the evaluation function (see above). The algorithm terminates when we cannot find a better-quality solution. The problem with hill climbing is that it can easily get stuck in a local optimum (see above). Consider Figure 1.1.

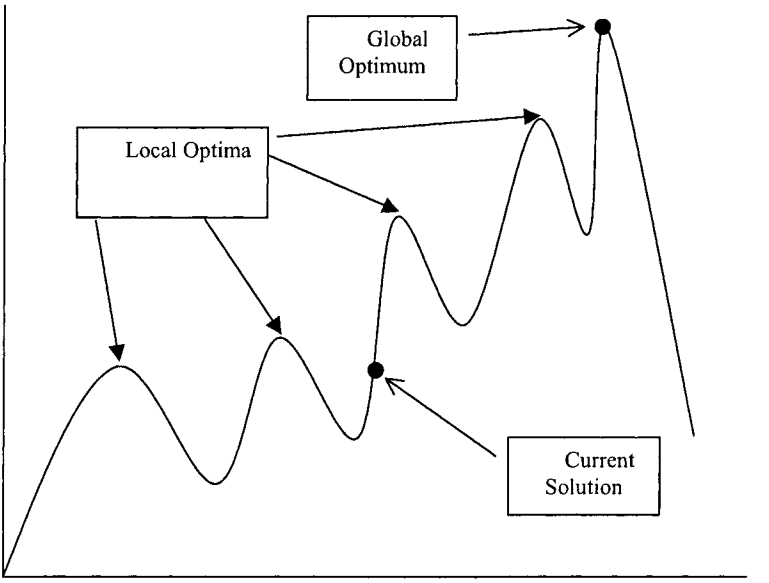

Figure 1.1. Hill Climbing getting stuck in a local optimum and the concept of local and global optima.

If the current solution is the one shown in Figure 1.1, then hill climbing will only be able to find one of the local optima shown (the one directly above it in this case). At that point, there will be no other better solutions in its neighborhood and the algorithm will terminate.

Both simulated annealing (Chapter 7) and tabu search (Chapter 6) are variations of hill climbing but they incorporate a mechanism to help the search escape from local optima.

Metaheuristics This term refers to a certain class of heuristic methods. Fred Glover first used it and he defines it (Glover, 1997) as follows:

> A meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality. The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule.

Osman and Kelly (1996) offer the following definition:

> A meta-heuristic is an iterative generation process which guides a subordinate heuristic ....

The study and development of metaheuristics has become an extremely important area of research into search methodologies. In common usage, in the

literature, the term tends to be used to refer to the broad collection of relatively *sophisticated* heuristic methods that include simulated annealing, tabu search, genetic algorithms, ant colony methods and others (all of which are discussed in detail in this book). The term is employed sometimes with and sometimes without the hyphen in the literature. It is also sometimes interchanged with the term "modern heuristics" (see Rayward-Smith et al. (1996). For more information about metaheuristics, see Glover and Kochenberger (2003), Osman and Kelly (1996), Voss et al. (1999), Ribeiro and Hansen (2002) and Resende and de Sousa (2004).

Evolutionary methods Evolutionary methods can be thought of as representing a subset of the metaheuristic approaches and are typified by the fact that they maintain a *population* of candidate solutions and these solutions compete for survival. Such approaches are inspired by evolution in nature.

Some of the methods in this book are evolutionary. Chapter 4 (Genetic Algorithms) represents perhaps the best known evolutionary approach but there are many others including genetic programming (Chapter 5) and ant algorithms (Chapter 14).

Hyper-heuristics Hyper-heuristics can be confused with metaheuristics but the distinction between the two terms is quite clear. Hyper-heuristics are simply methods which search through a search space of heuristics (or search methods). They can be defined as *heuristics to choose heuristics*. Most implementations of metaheuristics explore a search space of solutions to a given problem but they can be (and sometimes are) employed as hyper-heuristics. The term hyper-heuristic only tells you that we are operating on a search space of heuristics. It tells you nothing else. We may be employing a metaheuristic to do this search and we may not. The actual search space being explored may include metaheuristics and it may not (but very little work has actually been done which includes metaheuristics among the search space being addressed). Chapter 17 describes hyper-heuristics in more detail and readers are also referred to Burke et al. (2003).

1.4 SOURCES OF ADDITIONAL INFORMATION

This section provides a list of journals (in alphabetical order) across a range of disciplines that regularly publish papers upon aspects of decision support methodologies. This list is certainly not exhaustive. However, it provides a starting point for the new researcher and that is the sole purpose of presenting it here. We have purposefully not provided URL links to the journals as many will change after going to press, but an internet search for the journal title will quickly locate the home page.

- ACM Journal of Experimental Algorithmics
- Annals Of Operations Research
- Applied Artificial Intelligence
- Applied Intelligence
- Artificial Intelligence
- Artificial Life
- Computational Intelligence
- Computer Journal
- Computers & Industrial Engineering
- Computers & Operations Research
- Decision Support Systems
- Engineering Optimization
- European Journal Of Information Systems
- European Journal Of Operational Research
- Evolutionary Computation
- Fuzzy Sets And Systems
- Genetic Programming and Evolvable Machines
- IEEE Transactions On Computers
- IEEE Transactions On Evolutionary Computation
- IEEE Transactions On Fuzzy Systems
- IEEE Transactions On Neural Networks
- IEEE Transactions On Systems Man And Cybernetics Part A—Systems And Humans
- IEEE Transactions On Systems Man And Cybernetics Part B—Cybernetics
- IEEE Transactions On Systems Man And Cybernetics Part C—Applications And Review
- IIE Transactions
- INFORMS Journal On Computing

- Interfaces
- International Journal Of Systems Science
- International Transactions On Operational Research
- Journal Of Artificial Intelligence Research
- Journal Of Global Optimization
- Journal Of Heuristics
- Journal Of Optimization Theory And Applications
- Journal of Scheduling
- Journal Of The ACM
- Journal Of The Operational Research Society
- Knowledge-Based Systems
- Machine Learning
- Management Science
- Mathematical Programming: Series A and B
- Mathematics of Operations Research
- Neural Computation
- Neural Computing & Applications
- Neural Networks
- Neurocomputing
- Omega - International Journal of Management Science
- Operations Research
- Operations Research Letters
- OR Spectrum
- SIAM Journal on Computing
- SIAM Journal On Optimization

The following list of references just includes those volumes and papers which give an overview of search and optimization methodologies and some well studied search/optimization problems. More detailed bibliographies and sources of additional information are given at the end of each chapter throughout the book.

References

Aarts, E. and Lenstra, J. K. (eds), 1997, *Local Search in Combinatorial Optimization*, Wiley, New York.

Bronson, R. and Naadimuthu, G, 1997, Operations Research, Schaum's Outlines, 2nd edn, McGraw-Hill, New York.

Burke, E. K., De Causmaecker, P., Vanden Berghe, G. and Van Landeghem, R., 2004, The state of the art of nurse rostering, *J. Scheduling* **7**:441–499.

Burke, E. K., Kendall, G., Newall, J. P., Hart, E., Ross, P. and Schulenburg, S., 2003, Hyper-heuristics: An emerging direction in modern search technology, in: *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, eds, Chapter 16, pp. 457–474.

Burke, E. K., Petrovic, S., 2002, Recent research directions in automated timetabling, *Eur. J. Oper. Res.* **140**:266–280.

Carter, M. W., and Price, C. C., 2001, *Operations Research: A Practical Introduction*, CRC Press, Boca Raton, FL.

Callan R., 2003, *Artificial Intelligence*, Palgrave Macmillan, London.

Cawsey A., 1998, *The Essence of Artificial Intelligence*, Prentice-Hall, Englewood Cliffs, NJ.

Dell'Amico, M., Maffioli, F. and Martello, S. (eds), 1997, *Annotated Bibliographies in Combinatorial Optimization*, Wiley, New York.

Dowsland, K. A. and Dowsland, W. B., 1992, Packing problems, *Eur. J. Oper. Res.* **56**:2–14.

Gass, S. I. and Harris, C. M., 2001, *Encyclopaedia of Operations Research and Management Science*, Kluwer, Dordrecht.

Glover, F. and Kochenberger, G. (eds), 2003, *Handbook of Metaheuristics*, Kluwer, Dordrecht.

Glover, F. and Laguna, M., 1997, *Tabu Search*, Kluwer, Dordrecht.

Hillier F. S. and Liberman G. J., 2005, *Introduction to Operations Research*, McGraw-Hill, New York (8th edn).

Johnson, D. S. and McGeoch, L. A., 1997, The travelling salesman problem: A case study, in: E. Aarts and J. K. Lenstra, eds, *Local Search in Combinatorial Optimization*, Wiley, New York, pp. 215–310.

Kirby, M. W., 2003, *Operational Research in War and Peace: The British Experience from the 1930s to 1970*, Imperial College Press, London.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (eds), 1985, *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York (reprinted with subject index 1990).

Leung, J. Y-T. (ed.), 2004, *Handbook of Scheduling*, Chapman and Hall/CRC Press, Boca Raton, FL.

Luger G. F. A., 2002, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison-Wesley, Reading, MA, 4th edn.

McCarthy, J., 1996, *Defending AI Research: A Collection of Essays and Reviews*, CSLI Publications, Stanford, CA.

Michaelwicz, Z. and Fogel D. B., 2004, *How to Solve It: Modern Heuristics*, Springer, Berlin, 2nd edn.

Negnevitsky M., 2005, *Artificial Intelligence: A Guide to Intelligent Systems*, Addison-Wesley, Reading, MA, 2nd edn.

Nilsson, N., 1998, *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, San Mateo, CA.

Oxford Dictionary of Computing, 1997, Oxford University Press, Oxford, 4th edn.

Osman, I. H. and Kelly J. P. (eds), 1996, *Metaheuristics: Theory and Applications*, Kluwer, Dordrecht.

Pardalos, P. M. and Resende, M. G. C. (eds), 2002, *Handbook of Applied Optimization*, Oxford University Press, Oxford.

Petrovic, S. and Burke, E. K., 2004, University timetabling, Chapter 45 of J. Y-T. Leung, ed., *Handbook of Scheduling*, Chapman and Hall/CRC Press, Boca Raton, FL.

Rayward-Smith V. J., Osman I. H., Reeves C. R. and Smith G. D., 1996, *Modern Heuristic Search Methods*, Wiley, New York.

Reeves, C. R., 1996, Modern heuristic techniques, in: V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith, *Modern Heuristic Search Methods*, Wiley, New York, pp. 1–25.

Resende, M. G. C. and de Sousa, J. P. (eds), 2004, *Metaheuristics: Computer Decision Making*, Kluwer, Dordrecht.

Ribeiro, C. C. and Hansen, P. (eds), 2002, *Essays and Surveys in Metaheuristics*, Kluwer, Dordrecht.

Rich E. and Knight K., 1991, *Artificial Intelligence*, McGraw-Hill, New York, 2nd edn.

Russell, S. and Norvig, P., 2003, *Artificial Intelligence: A Modern Approach*, Prentice-Hall, Englewood Cliffs, NJ.

Schaerf, A., 1999, A survey of automated timetabling, *Artif. Intell. Rev.* **13**:87–127.

Taha H. A., 2002, *Operations Research: An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 7th edn.

Urry S., 1991, *An Introduction to Operational Research: The Best of Everything*, Longmans, London.

Voss, S, Martello, S., Osman, I. H. and Roucairol, C. (eds), 1999, *Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Dordrecht.

Winston, W. L., 2004, *Operations Research: Applications and Algorithms*, Duxbury Press, Philadelphia, PA, 4th edn.

Chapter 2

CLASSICAL TECHNIQUES

Kathryn A. Dowsland
Gower Optimal Algorithms Ltd
Swansea, UK
and
The School of Computer Science and IT
University of Nottingham, UK

2.1 INTRODUCTION

The purpose of this chapter is to provide an introduction to three classical search techniques, branch and bound, dynamic programming and network flow programming, all of which have a well established record in the solution of both classical and practical problems. All three have their origins in, or prior to, the 1950s and were the result of a surge in interest in the use of mathematical techniques for the solution of practical problems. The timing was in part due to developments in Operations Research in World War II, but was also spurred on by increasing competition in the industrial sector and the promise of readily accessible computing power in the foreseeable future. A fourth technique belonging to this class, that of Integer Programming, is covered in Chapter 3. Given their age, it is not surprising that they no longer generate the same level of excitement as the more modern approaches covered elsewhere in this volume, and as a result they are frequently overlooked. This effect is reinforced as many texts such as this omit them—presumably because they have already been covered by a range of sources aimed at a wide variety of different abilities and backgrounds. In this volume we provide an introduction to these well-established classics alongside their more modern counterparts. Although they have shortcomings, many of which the more recent approaches were designed to address, they still have a role to play both as stand-alone techniques and as important ingredients in hybridized solution methods.

The chapter is meant for beginners and it is possible to understand and use the techniques covered without any prerequisite knowledge. However, some of the examples in the chapter are based on problems in graph theory. In all cases, the problems and specialist terms are defined in full, but a basic knowledge of

graph theory terminology such as that provided in Balakrishnan (1997) would be useful. Some of the examples also belong to a class of problems known as *linear programs* (LPs) and some of the discussion in the section on network flows makes use of the relationship between network flow problems and linear programming problems. Although knowledge of LPs is not necessary to understand the algorithms or examples as these are all couched in purely combinatorial terms we start the chapter with a brief introduction to linear programming. Further details can be found in Anderson et al. (1997).

The chapter is organized as follows. The overview of LPs is followed by three sections introducing branch and bound, dynamic programming and network flow programming. In each case an introductory description is followed by two or more examples of their use in solving different problems, including a worked numerical example in each case. Each section ends with a brief discussion of more advanced issues. Section 2.6 looks at some problems that frequently occur as sub-problems in the solution of more complex problems and suggests algorithms based on the techniques covered in Sections 2.3–2.5 for their solution. Section 2.7 takes a brief look at potential future applications and Section 2.8 provides some hints and tips on how to get started with each of the techniques. Additional sources of information not covered in the references are given at the end of the chapter.

2.2 LINEAR PROGRAMMING

2.2.1 Introduction

This section provides a brief overview of those aspects of LPs that are relevant to the remainder of this chapter. We start by outlining the basic features of an LP model and then go on to look at an important concept of such models—that of duality. We do not go into any detail with regard to solution algorithms for two reasons. Firstly, they are not necessary in order to understand the material presented in the remainder of the chapter. Secondly, LP packages and solution code are available from a wide variety of sources so that it is no longer necessary for a potential user to develop their own solution code.

2.2.2 The Linear Programming Form

A linear programming problem is an optimization problem in which both the objective (i.e. the expression that is to be optimized) and the constraints on the solution can be expressed as a series of linear expressions in the decision variables. If the problem has n variables then the constraints define a set of hyper-planes in n-dimensional space. These are the boundaries of an n-dimensional region that defines the set of feasible solutions to the problem and

is known as the feasible region. The following example illustrates the form of a simple linear programming problem.

A Simple Example A clothing manufacturer makes three different styles of T-shirt. Style 1 requires 7.5 minutes of cutting time, 12 minutes of sewing time, 3 minutes of packaging time and sells at a profit of £3. Style 2 requires 8 minutes of cutting time, 9 minutes of sewing time, 4 minutes of packaging time and makes £5 profit. Style 3 requires 4 minutes of cutting time, 8 minutes of sewing time and 2 minutes of packaging time and makes £4 profit. The company wants to determine production quantities of each style for the coming month. They have an order for 1000 T-shirts of style 1 that must be met, and have a total of 10 000 minutes available for cutting, 18 000 minutes for sewing and 9000 minutes available for packaging. Assuming that they will be able sell as many T-shirts as they produce in any of the styles, how many of each should they make in order to maximize their profit?

We can formulate the problem mathematically as follows. First we define three decision variables x_1, x_2 and x_3 representing the number of T-shirts manufactured in styles 1, 2 and 3 respectively. Then the whole problem can be written as

$$\begin{aligned}
\text{maximize } & 3x_1 + 5x_2 + 4x_3 && && (2.1)\\
\text{subject to } & 7.5x_1 + 8x_2 + 4x_3 && \leq\ 10000 && (2.2)\\
& 12x_1 + 9x_2 + 8x_3 && \leq\ 18000 && (2.3)\\
& 3x_1 + 4x_2 + 2x_3 && \leq\ 9000 && (2.4)\\
& x_1 && \geq\ 1000 && (2.5)\\
& x_1, x_2, x_3 && \geq\ 0 && (2.6)
\end{aligned}$$

Expression (2.1) defines the profit. This is what we need to maximize and is known as the *objective function*. The remaining expressions are the *constraints*. Constraints (2.2)–(2.4) ensure that the hours required for cutting, sewing and packaging do not exceed those available. Constraint (2.5) ensures that at least 1000 T-shirts of style 1 are produced and constraint (2.6) stipulates that all the decision variables must be non-negative. Note that all the expressions are linear in the decision variables, and we therefore have a linear programming formulation of the problem.

The General LP Format In general a linear program is any problem of the form

$$\begin{aligned} \text{max(or min)} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{such that} \quad & \sum_{i=1}^{n} a_{1i} x_i \sim b_1 \\ & \vdots \\ & \sum_{i=1}^{n} a_{mi} x_i \sim b_m \end{aligned} \tag{2.7}$$

where $\sim$ is one of $\geq$, $=$ or $\leq$.

The important point about a linear programming model is that the feasible region is a convex space and the objective function is a convex function. Optimization theory therefore tells us that as long as the variables can take on any real non-negative values (possibly bounded above) then the optimal solution can be found at an extreme point of the feasible region. It is also possible to derive conditions that tell us whether or not a given extreme point is optimal. Standard LP solution approaches based on these observations can solve problems involving many thousands of variables in reasonable time. As we will see later in this chapter there are special cases where these techniques work even when the variables are constrained to take on integer or binary values. The general case where the variables are constrained to be integer, known as integer programming, is more difficult and is covered in Chapter 3.

Although it makes sense when formulating LPs to use the flexibility of the formulation above in allowing either a maximization or minimization objective and any combination of inequalities and equalities for the constraints, much linear programming theory (and therefore the solution approaches based on the theory) assume that the problem has been converted to a standard form in which the objective is expressed in terms of a maximization problem, all the constraints apart from the non-negativity conditions are expressed as equalities, all right-hand side values $b_1, \ldots, b_m$ are non-negative and all decision variables are non-negative. A general formulation can be converted into this form by the following steps.

1 If the problem is a minimization problem the signs of the objective coefficients $c_1, \ldots, c_n$ are changed.

2 Any variable not constrained to be non-negative is written as the difference between two non-negative variables.

3 Any constraint with a negative right-hand side is multiplied by -1.

4 Any constraint which is an inequality is converted to an equality by the introduction of a new variable, (known as a slack variable) to the left-hand side. The variable is added in the case of a $\leq$ constraint and subtracted in the case of a $\geq$ constraint. The formulation is often written in matrix form:

$$\begin{aligned} \max \quad & CX \\ \text{s.t.} \quad & AX = b \\ & X \geq 0 \end{aligned} \tag{2.8}$$

where $C = (c_1, \ldots, c_n)$, $b = (b_1, \ldots, b_m)^{\mathrm{T}}$, $X = (x_1, \ldots, x_n)^{\mathrm{T}}$ and $A = (a_{ij})_{m \times n}$.

2.2.3 Duality

An important concept in linear programming is that of *duality*. For a maximization problem in which all the constraints are $\leq$ constraints and all variables are non-negative, i.e. a problem of the form

$$\begin{aligned} \max \quad & CX \\ \text{s.t.} \quad & AX \leq b \\ & X \geq 0 \end{aligned} \tag{2.9}$$

the dual is defined as

$$\begin{aligned} \min \quad & b^{\mathrm{T}}Y \\ \text{s.t.} \quad & A^{\mathrm{T}}Y \geq C^{\mathrm{T}} \\ & Y \geq 0 \end{aligned} \tag{2.10}$$

The original problem is known as the *primal*. Note that there is a dual variable y_i associated with each constraint in the primal problem and a dual constraint associated with each variable x_i in the primal. (The dual of a primal with a more general formulation can be derived by using rules similar to those used to convert a problem into standard form to convert the primal into the above form, with equality constraints being replaced by the equivalent two inequalities.)

The dual has the important property that the value of the objective in the optimal solution to the primal problem is the same as the optimal solution for the dual problem. Moreover, the *theorem of complementary slackness* states that if both dual and primal have been converted to standard form, with $s_1, \ldots, s_m$ the slack variables in the primal problem and $e_1, \ldots, e_n$ the slack variables in the dual, then if $X = (x_1, \ldots, x_n)$ is feasible for the dual and $Y = (y_1, \ldots, y_m)$ is feasible for the primal, X and Y are optimal solutions to the primal and dual

respectively if and only if $s_i y_i = 0, \forall i = 1, m$ and $e_j x_j = 0, \forall j = 1, n$. This relationship is an important feature in the specialist solution algorithm for the minimum cost network flow algorithm presented later in this chapter and underpins other LP-based solution approaches.

2.2.4 Solution Techniques

Solution approaches for linear programming fall into two categories. Simplex-type methods search the extreme points of the feasible region of the primal or the dual problem until the optimality conditions are satisfied. The technique dates from the seminal work of Dantzig (1951). Since then the basic technique has been refined in a number of ways to improve the overall efficiency of the search and to improve its operation on problems with specific characteristics. Variants of the simplex approach are available in a number of specialist software packages, as a feature of some spreadsheet packages, and as freeware from various web-based sources.

Although they perform well in practice, simplex-based procedures suffer from the disadvantage that they have poor worst-case time-performance guarantees. This deficiency led to the investigation of interior point methods, so called because they search a path of solutions through the interior of the feasible region in such a way as to arrive at an optimal point when they hit the boundary. Practical interior point methods can be traced back to the work of Kamarkar (1984), although Khachiyan's (1979) ellipsoid method was the first LP algorithm with a polynomial time guarantee. Recent interior point methods have proved efficient, in particular for large LPs, and there has also been some success with hybridizations of interior point and simplex approaches. While the choice of solution approach may be important for very large or difficult problems, for most purposes standard available code based on simplex-type approaches should suffice.

2.3 BRANCH AND BOUND

2.3.1 Introduction

When faced with the problem of finding an optimum over a finite set of alternatives an obvious approach is to enumerate all the alternatives and then select the best. However, for anything other than the smallest problems such an approach will be computationally infeasible. The rationale behind the branch and bound algorithm is to reduce the number of alternatives that need to be considered by repeatedly partitioning the problem into a set of smaller sub-problems and using local information in the form of bounds to eliminate those that can be shown to be sub-optimal. The simplest branch and bound implementations are those based on a constructive approach in which partial solutions are built

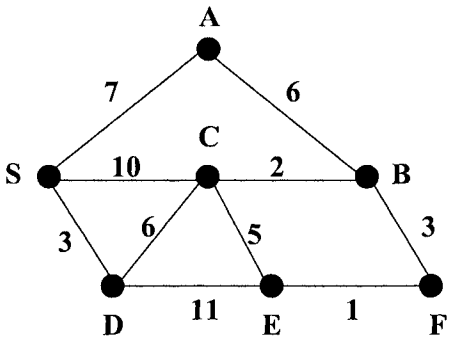

Figure 2.1. Shortest path network.

up one element at a time. We therefore start by introducing the technique in this context before going on to show how it can be extended to its more general form.

Assume that we have a problem whose solutions consist of finite vectors of the form $(x_1, x_2, \ldots, x_k)$ where k may vary from solution to solution. Those combinations of values that form feasible vectors are determined by the problem constraints. The set of all possible solutions can be determined by taking each feasible value for x_1, then for each x_1 considering all compatible values for x_2, then for each partial solution $(x_1, x_2, \ldots)$ considering all compatible x_3 etc. This process can be represented as a tree in which the branches at level i correspond to the choices for x_i given the choices already made for $x_1, \ldots, x_{i-1}$, and the nodes at level i correspond to the partial solutions of the first i elements. This is illustrated with reference to Figures 2.1 and 2.2. Figure 2.2 is the tree enumerating all simple routes (i.e. routes without loops) from S to F in the network shown in Figure 2.1. The branches at level 1 represent all the possibilities for leaving S and the branches at lower levels represent all the possibilities for extending the partial solution represented by the previous branches by one further link. The node at the top of the tree is sometimes referred to as the *root*, and the relationship between a given node and one immediately below is sometimes referred to as *parent/child* or *father/son*. All nodes that can be reached from the current node by traveling down the tree are referred to as *descendants* and nodes without any descendants are *terminal nodes*.

If we wish to search the tree in an ordered fashion we need to define a set of rules determining the order in which the branches are to be explored. This is known as the *branching strategy*. The two simplest strategies are known as *depth-first search* and *breadth-first search*. Depth-first search (also known as branch and backtrack) moves straight down a sequence of branches until